**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Дисциплина «Объектно-ориентированное проектирование и программирование»

ОТЧЕТ

Лабораторная работа № 1

Основы синтаксиса объектно-ориентированного языка программирования

Выполнил студент

Группы ИТД-21

Чайдаков И.М.

Принял преподаватель

Башаримов Ю. С.

Гомель 2024

Основы синтаксиса объектно-ориентированного языка программирования

**Цель работы:** научиться писать программы используя инструменты объектно-ориентированного языка программирования *C#*. Научиться работать с классами и их методами.

**Задание:** Необходимо разработать класс представляющий собой параллелограмм. Фигура должна задаваться координатами ее вершин. Должны быть реализованы функции проверки возможности существования фигуры, вычисления ее параметров и проверяющие принадлежность заданной точки к фигуре.

**Ход выполнения задания:**

**1.** Описываем класс представляющий собой координату на плоскости (см. Приложение А).

**2.** Описываем класс представляющий собой параллелограмм и описываем ему все необходимые функции (см. Приложение А).

**3.** Описываем класс *main* в котором создаем пользовательский интерфейс для взаимодействия с параметрами фигуры (см. Приложение B).

**4.** Запускает файл класса main и видим перед собой консольный интерфейс (Рисунок 1).

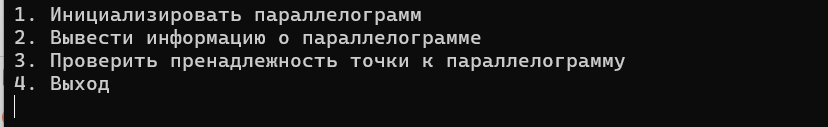


Рисунок 1 — Интерфейс программы.

**5.** Инициализируем параллелограмм на вершинах {(1, 5), (7, 5), (-2, 0), (4, 0)} и выводим о нем информацию (Рисунок 2).

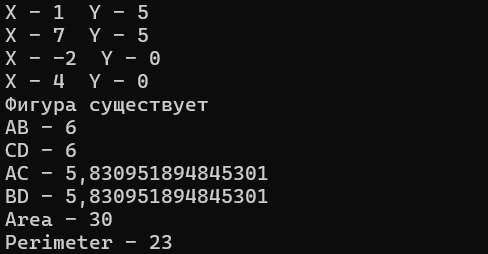


Рисунок 2 — Информация о созданном параллелограмме.

**6.** Выполняем проверки принадлежности заданной точки (2, 3) к фигуре (Рисунок 3).

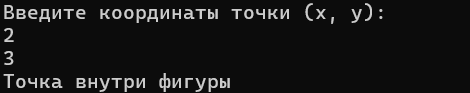


Рисунок 3 — Проверка на принадлежность точки к фигуре.

Как мы можем наблюдать, программы исправно выполняет все свои функции. Задание выполнено.

**Вывод:** Объектно-ориентированное программирование позволяет создавать классы с данными и методами которые позже можно импортировать в другие программы. Такой подход значительно повышает читаемость кода и возможность его редактирования так как, по сути, программа состоит из комбинации независимых блоков.

**ПРИЛОЖЕНИЕ А**

**Код используемых классов**

**Vertex**

namespace Library

{

    /// <summary>

    /// класс, представляющий вершину из X и Y координат.

    /// </summary>

    /// <param name="xCord">X-координата вершины.</param>

    /// <param name="yCord">Y-координата вершины.</param>

    public class Vertex(int xCord, int yCord)

    {

        /// <summary>

        /// X-координата вершины.

        /// </summary>

        public int xCord = xCord;

        /// <summary>

        /// Y-координата вершины.

        /// </summary>

        public int yCord = yCord;

        /// <summary>

        /// вывод информации о вершинаъ.

        /// </summary>

        public void Print()

        {

            Console.WriteLine($"X - {xCord}  Y - {yCord}");

        }

    }

}

**Parallelogram**

namespace Library

{

    /// <summary>

    /// Представляет параллелограмм, определенный четырьмя вершинами и различными характеристиками, такими как площадь, периметр и длины сторон.

    /// </summary>

    public class Parallelogram(Vertex a, Vertex b, Vertex c, Vertex d)

    {

        /// <summary>

        /// Первая вершина параллелограмма.

        /// </summary>

        public Vertex a = a;

        /// <summary>

        /// Вторая вершина параллелограмма.

        /// </summary>

        public Vertex b = b;

        /// <summary>

        /// Третья вершина параллелограмма.

        /// </summary>

        public Vertex c = c;

        /// <summary>

        /// Четвертая вершина параллелограмма.

        /// </summary>

        public Vertex d = d;

        /// <summary>

        /// Длина стороны AB.

        /// </summary>

        public double ab;

        /// <summary>

        /// Длина противоположной стороны CD.

        /// </summary>

        public double cd;

        /// <summary>

        /// Диагональ AC.

        /// </summary>

        public double ac;

        /// <summary>

        /// Диагональ BD.

        /// </summary>

        public double bd;

        /// <summary>

        /// Площадь параллелограмма.

        /// </summary>

        public double area;

        /// <summary>

        /// Высота параллелограмма.

        /// </summary>

        public int height;

        /// <summary>

        /// Периметр параллелограмма.

        /// </summary>

        public int perimeter;

        /// <summary>

        /// Проверяет, существует ли параллелограмм, основываясь на расположении вершин.

        /// </summary>

        /// <returns>Возвращает true, если параллелограмм существует; иначе - false.</returns>

        public bool ItExist()

        {

            if (a.yCord == b.yCord && c.yCord == d.yCord)

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        /// <summary>

        /// Вычисляет длины сторон и диагоналей параллелограмма на основе координат вершин.

        /// </summary>

        public void FindSideLengths()

        {

            ab = b.xCord - a.xCord;

            cd = d.xCord - c.xCord;

            ac = Math.Pow(Math.Pow((double)Math.Abs(a.xCord - c.xCord), (double)2) + Math.Pow((double)(a.yCord - c.yCord), (double)2), (double)0.5);

            bd = Math.Pow(Math.Pow((double)Math.Abs(b.xCord - d.xCord), (double)2) + Math.Pow((double)(b.yCord - d.yCord), (double)2), (double)0.5);

        }

        /// <summary>

        /// Вычисляет площадь параллелограмма.

        /// </summary>

        public void FindArea()

        {

            height = a.yCord - c.yCord;

            area = height \* cd;

        }

        /// <summary>

        /// Вычисляет периметр параллелограмма.

        /// </summary>

        public void FindPerimeter()

        {

            perimeter = (int)(ab + cd + ac + bd);

        }

        /// <summary>

        /// Определяет, находится ли заданная точка внутри параллелограмма.

        /// </summary>

        /// <param name="xCord">X-координата точки.</param>

        /// <param name="yCord">Y-координата точки.</param>

        public void DotExam(int xCord, int yCord)

        {

            bool[] vectors = new bool[4];

             int FindVector(int x0, int y0, int xA, int yA, int xB, int yB)

            {

                return (xA - x0) \* (yB - y0) - (yA - y0) \* (xB - x0);

            }

            vectors[0] = FindVector(xCord, yCord, a.xCord, a.yCord, b.xCord, b.yCord) > 0;

            vectors[1] = FindVector(xCord, yCord, b.xCord, b.yCord, d.xCord, d.yCord) > 0;

            vectors[2] = FindVector(xCord, yCord, d.xCord, d.yCord, c.xCord, c.yCord) > 0;

            vectors[3] = FindVector(xCord, yCord, c.xCord, c.yCord, a.xCord, a.yCord) > 0;

            bool isInside = (vectors[0] == vectors[1]) && (vectors[1] == vectors[2]) && (vectors[2] == vectors[3]);

            if (isInside)

            {

                Console.WriteLine("Точка внутри фигуры");

            }

            else

            {

                Console.WriteLine("Точка вне фигуры");

            }

        }

        /// <summary>

        /// Выводит информацию о вершинах, сторонах, площади и периметре параллелограмма.

        /// </summary>

        public void Print()

        {

            a.Print();

            b.Print();

            c.Print();

            d.Print();

            if (this.ItExist())

            {

                Console.WriteLine("Фигура существует");

            }

            else

            {

                Console.WriteLine("Фигура не существует");

            }

            Console.WriteLine($"AB - {ab}\nCD - {cd}\nAC - {ac}\nBD - {bd}");

            Console.WriteLine($"Area - {area}");

            Console.WriteLine($"Perimeter - {perimeter}");

        }

    }

}

**ПРИЛОЖЕНИЕ**

**Код класса *main***

#pragma warning disable CS8602 // Разыменование вероятной пустой ссылки.

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

using Library;

/// <summary>

/// Главный класс для работы с параллелограммом.

/// </summary>

class Program

{

    /// <summary>

    /// Входная точка программы.

    /// </summary>

    static void Main()

    {

        string?[] input = new string?[8];

        int count;

        int endIsNear = 0;

        int userChoice;

        Vertex aVert = new Vertex(0, 0);

        Vertex bVert = new Vertex(0, 0);

        Vertex cVert = new Vertex(0, 0);

        Vertex dVert = new Vertex(0, 0);

        Parallelogram parall = new Parallelogram(aVert, bVert, cVert, dVert);

        while (endIsNear == 0)

        {

            count = 0;

            Console.WriteLine("1. Инициализировать параллелограмм");

            Console.WriteLine("2. Вывести информацию о параллелограмме");

            Console.WriteLine("3. Проверить принадлежность точки к параллелограмму");

            Console.WriteLine("4. Выход");

            input[0] = Console.ReadLine();

            if (input != null)

            {

                userChoice = int.Parse(input[0]);

                switch (userChoice)

                {

                    case 1:

                        {

                            Console.WriteLine("Введите координаты вершин параллелограма (x1, y1, x2, y2, x3, y3, x4, y4): ");

                            for (count = 0; count < 8; count++)

                            {

                                input[count] = Console.ReadLine() ?? "";

                            }

                            if (input != null)

                            {

                                aVert = new Vertex(int.Parse(input[0]), int.Parse(input[1]));

                                bVert = new Vertex(int.Parse(input[2]), int.Parse(input[3]));

                                cVert = new Vertex(int.Parse(input[4]), int.Parse(input[5]));

                                dVert = new Vertex(int.Parse(input[6]), int.Parse(input[7]));

                                parall = new Parallelogram(aVert, bVert, cVert, dVert);

                            }

                            else

                            {

                                Console.WriteLine("Ошибка: Введено значение null.");

                            }

                            count = 0;

                            break;

                        }

                    case 2:

                        {

                            parall.FindSideLengths();

                            parall.FindPerimeter();

                            parall.FindArea();

                            parall.Print();

                            break;

                        }

                    case 3:

                        {

                            Console.WriteLine("Введите координаты точки (x, y): ");

                            for (count = 0; count < 2; count++)

                            {

                                input[count] = Console.ReadLine() ?? "";

                            }

                            if (input != null)

                            {

                                parall.DotExam(int.Parse(input[0]), int.Parse(input[1]));

                            }

                            else

                            {

                                Console.WriteLine("Ошибка: Введено значение null.");

                            }

                            count = 0;

                            break;

                        }

                    case 4:

                        {

                            endIsNear = 1;

                            break;

                        }

                    default:

                        {

                            Console.WriteLine("Ошибка: Введено некорректное значение.");

                            break;

                        }

                }

            }

            else

            {

                Console.WriteLine("Ошибка: Введено значение null.");

            }

        }

    }

}